Higher-dimensional antiderivatives and the efficient computation of electrostatic potentials

Ulrich Mutze
www.ulrichmutze.de

July 26, 2010

It is shown how the fundamental theorem of calculus for several variables can be used for efficiently computing the electrostatic potential of moderately complicated charge distributions. This is exemplified with a charge distribution which resembles the letter F in shape.

1 Introduction

As it was shown in [1], the integral of a function [1] f over a *generalized rectangle* [2] $R \subset \mathbb{R}^n$ can be expressed as a simple linear combination of values which an antiderivative F of f takes at the corners of R. Here the antiderivative can equivalently be defined as any cumulative distribution function

$$F(x_1,\ldots,x_n) = \int_{a_n}^{x_n} \big(\ldots \big(\int_{a_1}^{x_1} f(y_1,\ldots,y_n)\, \mathrm{d}y_1 \big) \ldots \big)\, \mathrm{d}y_n \tag{1}$$

or as any function F satisfying

$$\frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_n} F(x_1,\ldots,x_n) = f(x_1,\ldots,x_n)\,. \tag{2}$$

This constitutes a direct generalization of the *fundamental theorem of calculus* from a single real variable to n such variables. For the linear combination mentioned above the following notation has been introduced [2]

$$\int_R f(x_1,\ldots,x_n)\,\mathrm{d}x_1 \ldots \mathrm{d}x_n \;=\; \sum_{c\in\nabla R} \alpha_R(c)\, F(c)\,. \tag{3}$$

Here ∇R is the set of 2^n corners of R and the term $\alpha_R(c)$, which I will call the *weight* of corner c, takes values in $\{-1,+1\}$. These weights satisfy

$$\sum_{c\in\nabla R} \alpha_R(c) \;= 0\,. \tag{4}$$

[1] In this article we do not specify the regularity properties of the functions involved. One consistent setting is to assume that all functions to be integrated over some domain are continuous there.

[2] Special names are interval, rectangle, cuboid for $n = 1,2,3$ respectively. We will omit the adjective 'generalized' if the context makes clear which value of n is under consideration.

Equation (3) can easily be seen to persist if R is replaced by any finite *non-overlapping* [3] union of *compatible* [4] rectangles. The only difference is that now the weights may take integer values different from $+1, -1$. In this form, this relation has been given the name *discrete Green's theorem* [3], which also found its way into Wikipedia.

We need to catch up on defining α_R for the case that R is a rectangle in $\mathbb{R}^n$. We assume that R is not void. Then we have two 'main corners' $a, b \in \mathbb{R}^n$ such that $a_1 < b_1, \ldots, a_n < b_n$ and all corners can be represented as

$$\nabla R = \{(a_1\varepsilon_1 + b_1(1-\varepsilon_1), \ldots, a_n\varepsilon_n + b_n(1-\varepsilon_n)) \mid (\varepsilon_1, \ldots, \varepsilon_n) \in \{0,1\}^n\} . \tag{5}$$

Then the weight of the generic corner is given by

$$\alpha_R((a_1\varepsilon_1 + b_1(1-\varepsilon_1), \ldots, a_n\varepsilon_n + b_n(1-\varepsilon_n))) = (-1)^{\varepsilon_1+\ldots+\varepsilon_n} . \tag{6}$$

The present work carries out the electrostatic application which in [1] was mentioned as the motivation for investigating fast evaluation of integrals over rectangular domains. It is interesting to notice that in our case the essential efficiency gain comes from being able to compute antiderivatives in the sense of (2), whereas the newer applications of (3) seem to deal with analyzing digital images and get their efficiency gain from forming the discrete analog of (1) (for $n = 2$) for an image once, and then using it many times for the analysis of varying image regions.

2 Combining the contributions of compatible rectangles

On the right-hand side of equation (3) the domain of integration R enters as a set

$$\widehat{R} := \{(\alpha(c), c) \mid c \in \nabla R\} \tag{7}$$

of 2^n elements of $\mathbb{Z} \times \mathbb{R}^n$ and thus as a finite subset of $\mathbb{Z} \times \mathbb{R}^n$. Let us denote, as usual, by $\mathcal{P}(X)$ the set of all subsets of any set X, and by $\mathcal{P}_\mathrm{f}(X)$ the set of all finite such subsets. Then it is the element $\widehat{R} \in \mathcal{P}_\mathrm{f}(\mathbb{Z} \times \mathbb{R}^n)$ which (together with F) determines the expression that forms the right-hand side of equation (3). This expression can easily be generalized: for any $\Gamma \in \mathcal{P}_\mathrm{f}(\mathbb{Z} \times \mathbb{R}^n)$ and any real-valued function F on $\mathbb{R}^n$ we define the real number $\Gamma(F)$ by

$$\Gamma(F) := \sum_{p \in \Gamma} p_1 F(p_2) . \tag{8}$$

This gives equation (3) the concise form

$$\int_R f(x_1, \ldots, x_n)\,\mathrm{d}x_1 \ldots \mathrm{d}x_n \;=\; \int_R \frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_n} F(x_1, \ldots, x_n)\,\mathrm{d}x_1 \ldots \mathrm{d}x_n \;=\; \widehat{R}(F) . \tag{9}$$

It will turn out convenient to have a name for elements of $\mathcal{P}_\mathrm{f}(\mathbb{Z} \times \mathbb{R}^n)$: Let us call any such element a *corner complex*. If the corner complex results from a single rectangle R, and thus is of the form $\widehat{R}$, we call it a *simple corner complex*. Figure 1 shows three simple corner complexes for $n = 2$. Actually, only the points together with the accompanying numbers constitute the corner complex. The connecting lines symbolize the corresponding rectangle. A non-simple corner complex with 15 corners appears in Figure 2.

[3] This means that no intersection of such rectangles contains a non-void open set.
[4] This means that they all have the same dimension n and are axis-parallel to each other.

A natural question to ask is whether for all $\Gamma_1, \Gamma_2 \in \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n)$ there is a $\Gamma_3 \in \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n)$ such that for every real-valued function F on $\mathbb{R}^n$ we have

$$\Gamma_3(F) = \Gamma_1(F) + \Gamma_2(F) \,. \tag{10}$$

The answer is yes and we can define an operation

$$\uplus \,:\, \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n) \times \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n) \;\to\; \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n)$$

such that such a Γ_3 is given as $\Gamma_1 \uplus \Gamma_2$. The definition of this operation suggests itself: First we define an equivalence relation on $\mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n)$ by

$$(w,x) \cong (w',x') \quad \text{iff} \quad x = x' \,. \tag{11}$$

To define $\Gamma_1 \uplus \Gamma_2$ we first form $\Gamma_1 \cup \Gamma_2 \in \mathcal{P}_{\mathrm{f}}(\mathbb{Z} \times \mathbb{R}^n)$. With respect to the relation $\cong$ this set falls into equivalence classes. To each equivalence class we define a single element $(s,x) \in \mathbb{Z} \times \mathbb{R}^n$ where x is the second component of any member of the class (all these members have the same second component by definition of $\cong$) and s is the sum over all first components of elements in the class. From the set of all the elements (s,x) obtained in this way we eliminate the elements for which $s = 0$. The remaining elements constitute $\Gamma_1 \uplus \Gamma_2$ by definition. Since the operation combines forming set unions and adding numbers, the symbol $\uplus$ really fits. A typical situation leading to elimination of corners is symbolized in Figure 1.

Let $(R_1, \ldots, R_m)$ be a finite list of non-overlapping compatible rectangles and let $(\Gamma_1, \ldots, \Gamma_m)$ be the list of the corresponding simple corner complexes. Then the corner complex

$$\Gamma := \Gamma_1 \uplus \ldots \uplus \Gamma_m \tag{12}$$

has the property that $\Gamma(F)$ equals the value of the integral of $\frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_n} F$ over the domain

$$D := R_1 \cup \ldots \cup R_m \,. \tag{13}$$

Let us re-state this as an equation:

$$\boxed{\int_{R_1 \cup \ldots \cup R_m} f(x_1, \ldots, x_n)\, \mathrm{d}x_1 \ldots \mathrm{d}x_n = (\widehat{R}_1 \uplus \ldots \uplus \widehat{R}_m)(F)} \tag{14}$$

for all $R_1, \ldots, R_m$ as introduced above and for any antiderivative F of f. This is a direct generalization of (9). It is remarkable from a conceptual viewpoint and also from a computational one. Several remarks are in order. Some of them are to substantiate this claim.

1. The operation '$\uplus$' does not assume that the corner complexes are generated by starting from axis-parallel rectangles. If we would rotate one such rectangle a bit out of its initially fitting direction and leave the weights unchanged, we should not expect (14) to hold. A corresponding phenomenon can already be observed with a single rectangle and (3). If no assumption concerning overlap is made, the equation remains valid in a natural sense: an area belonging to more than one rectangle delivers more than one contributions to the integral. If the function f had an interpretation as a mass density attributed to the rectangles, this behavior would correspond to a kind of matter which allows the rectangles to penetrate each other without disturbance, so that the overlap of two rectangles has twice the matter density than a single rectangle.

2. It is essential that the operation '$\uplus$' is defined for general corner complexes and not only for simple ones. Already $\widehat{R}_1 \uplus \widehat{R}_2$ may no longer be simple and nevertheless we need to form $(\widehat{R}_1 \uplus \widehat{R}_2) \uplus \widehat{R}_3$.

3. The successive 'addition' of simple corner complexes is very simple to program. For each corner of the new rectangle R we have to see whether it is already present in the complex to which we have to 'add' R. If not, we add the corner together with the weight which it has in R. If it is already there, then it already has a weight, and we only have to add to this the weight coming from R.

4. It is instructive to compare this approach with the one which seems most natural at a first glance, and that seems to be the only one used so far in the literature: One considers the list $(R_1,\ldots,R_m)$ as the input to an algorithm which outputs the set of corners together with their proper weights. See Figure 1 in [2] where the occurrence of weights $\in\{-2,-1,1,2\}$ is classified in terms of geometrical aspects of the corner. In [3] the weight factors are expressed as limits which play a natural role in an extended calculus developed there. I know of no attempt to treat the case $n>2$ that way.

5. In the present treatment, the corners and the weights are in the game from the very beginning and they have only to be updated with each new rectangle to join the integration domain. This is one example more for the often-made observation that a good match between data types (here corner complexes) and algorithms (here $\uplus$) is the key to clear and efficient programs.

6. If the union of rectangles $R_1,\ldots,R_m$ forms a compact body with a well-defined surface, all corners belong to this surface so that the right-hand side of equation (14) is a linear combination of function values which F takes on the surface. Therefore it is interesting to see that also the *general Stokes formula* gives such a representation: Let us define the alternating $(n-1)$-form

$$\omega := \frac{1}{n}\sum_{i=1}^{n}(-1)^{i-1}\prod_{j=1,j\neq i}^{n}\frac{\partial}{\partial x_j}\;F\bigwedge_{j=1,j\neq i}^{n}\mathrm{d}x_j \tag{15}$$

then we have

$$\mathrm{d}\,\omega = \left(\frac{\partial}{\partial x_1}\ldots\frac{\partial}{\partial x_n}F\right)\;\mathrm{d}x_1\wedge\ldots\wedge\mathrm{d}x_n\,. \tag{16}$$

Then Stokes' theorem says for suitable subsets $\Omega\subset\mathbb{R}^n$ with surface $\partial\Omega$

$$\int_\Omega f(x_1,\ldots,x_n)\,\mathrm{d}x_1\ldots\mathrm{d}x_n = \int_\Omega \mathrm{d}\,\omega = \int_{\partial\Omega}\omega\,. \tag{17}$$

One easily shows that $\int_{\partial R}\omega$ and $\widehat{R}(F)$ coincide for any rectangle R and it is very plausible that also

$$\int_{\partial(R_1\cup\ldots\cup R_m)}\omega = (\widehat{R}_1\uplus\ldots\uplus\widehat{R}_m)(F)$$

for R_i as in (14). An interesting special case is the representation of the volume in terms of a surface integral:

$$\begin{aligned} f(x_1,\ldots,x_n) &:= 1\\ F(x_1,\ldots,x_n) &= x_1\ldots x_n\\ \omega &= \frac{1}{n}\sum_{i=1}^{n}(-1)^{i-1}x_i\bigwedge_{j=1,j\neq i}^{n}\mathrm{d}x_j\\ \int_\Omega \mathrm{d}x_1\ldots\mathrm{d}x_n &= \int_{\partial\Omega}\omega\,. \end{aligned} \tag{18}$$

3 Closed formulas for the electrostatic potential of rectangular sources

The electrostatic potential of distributed charges will be computed by means of antiderivatives of the Coulomb potential $1/r$ and equation (3). In equation (18) the antiderivative of the constant $\{1\}$-valued function is given for any value of n. The antiderivative of the Coulomb potential is much more complicated and results for general n seem to be beyond reach. Here is all we need for the intended electrostatic application:

The following three real-valued functions on $\mathbb{R}^3$

$$\begin{aligned}
\Phi_1(x,y,z) &:= \log(x+r)\,, \quad \text{where} \quad r := \sqrt{x^2+y^2+z^2}\,,\\
\Phi_2(x,y,z) &:= x\log(y+r)+y\log(x+r)-z\arctan\frac{xy}{zr}\,,\\
\Phi_3(x,y,z) &:= yz\log(x+r)+zx\log(y+r)+xy\log(z+r)\\
&\quad -\frac{x^2}{2}\arctan\frac{yz}{xr}-\frac{y^2}{2}\arctan\frac{zx}{yr}-\frac{z^2}{2}\arctan\frac{xy}{zr}
\end{aligned} \tag{19}$$

are antiderivatives of $1/r$ with respect to as many variables as their index says. Precisely

$$\frac{\partial}{\partial x}\Phi_1(x,y,z)=\frac{\partial}{\partial x}\frac{\partial}{\partial y}\Phi_2(x,y,z)=\frac{\partial}{\partial x}\frac{\partial}{\partial y}\frac{\partial}{\partial z}\Phi_3(x,y,z)=\frac{1}{\sqrt{x^2+y^2+z^2}}\,. \tag{20}$$

The validity of (20) can easily be verified using e. g. *Mathematica*. Finding the expression Φ_3 from the primary output of *Mathematica 7* required much non-trivial manual interference, just as finding the expression Φ_2 required such interference when using *Mathematica 3.0* (see [1]). Since r is a symmetric function of its three variables it is a natural requirement that Φ_2 should be symmetric with respect to an interchange of x and y and that Φ_3 should be symmetric with respect to any permutation of the variables. The expressions given in (19) satisfy these conditions. Achieving this involved some trial and error. I'm confident that the antiderivatives given here are the most simple ones possible.

Let us now describe in detail how the formulas (19) can be used to compute the electrostatic potential of a charge distribution [5]. Although our motivation was about areal charges (surface charges) we will now use volume charges as a pattern for the general case (linear charges, areal charges, spatial charges). We are interested in the integral

$$\varphi(p)=\int_G \frac{\rho(q)}{|p-q|}\,\mathrm{d}^3q\,. \tag{21}$$

Let us assume that the charge density ρ is constant in the domain G. Then the integral of interest is

$$\int_G \frac{1}{|p-q|}\,\mathrm{d}^3q =: \int_G K(p,q)\,\mathrm{d}^3q\,. \tag{22}$$

Considering p as fixed during the q-integration, we see that the task is to integrate the function $f_p : q \mapsto K(p,q)$ over domain G. From our previous discussion it is natural to look for an antiderivative of f_p i. e. for a function F_p which satisfies

$$\frac{\partial}{\partial q_1}\frac{\partial}{\partial q_2}\frac{\partial}{\partial q_3}F_p(q)=f_p(q)=K(p,q)=\frac{1}{|p-q|}\,. \tag{23}$$

[5] Here we deal with numbers and not with physical quantities. As a physical application we could also speak of the gravitational potential of a mass distribution.

To find a solution to this equation it helps to make a substitution $q - p =: s$. Since p is constant in the process, we have $\frac{\partial}{\partial q_i} = \frac{\partial}{\partial s_i}$ and thus

$$\frac{\partial}{\partial s_1}\frac{\partial}{\partial s_2}\frac{\partial}{\partial s_3}F_p(p+s) = \frac{1}{|s|}\,. \tag{24}$$

Writing $s =: (x,y,z)$ and $p =: (\hat{x},\hat{y},\hat{z})$ this says

$$\frac{\partial}{\partial x}\frac{\partial}{\partial y}\frac{\partial}{\partial z}F_{(\hat{x},\hat{y},\hat{z})}(\hat{x}+x,\hat{y}+y,\hat{z}+z) = \frac{1}{\sqrt{x^2+y^2+z^2}}\,. \tag{25}$$

According to (19) one solution is

$$F_{(\hat{x},\hat{y},\hat{z})}(\hat{x}+x,\hat{y}+y,\hat{z}+z) = \Phi_3(x,y,z) \tag{26}$$

or

$$F_{(\hat{x},\hat{y},\hat{z})}(x,y,z) = \Phi_3(x-\hat{x},y-\hat{y},z-\hat{z})\,. \tag{27}$$

This allows us to give an exact representation of the integral (22) for the special case that G is a cuboid and, hence, ∇G the set of its 8 corners. Equations (25), (3), (27) say

$$\begin{aligned}\varphi_G(\hat{x},\hat{y},\hat{z}) &:= \int_G \frac{\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z}{\sqrt{(\hat{x}-x)^2+(\hat{y}-y)^2+(\hat{z}-z)^2}}\\ &= \sum_{x\in\nabla G}\alpha_G(x)\,F_{(\hat{x},\hat{y},\hat{z})}(x,y,z)\\ &= \sum_{x\in\nabla G}\alpha_G(x)\,\Phi_3(x-\hat{x},y-\hat{y},z-\hat{z})\end{aligned} \tag{28}$$

and thus gives the electrostatic potential of a uniformly charged cuboid as a rather manageable explicit expression. This is even more true for charge distributions of lower dimensionality for which the corresponding expressions are as follows:

For a rectangular patch R (located in the plane $z = 0$) of uniform surface charge we have

$$\begin{aligned}\varphi_R(\hat{x},\hat{y},\hat{z}) &:= \int_R \frac{\mathrm{d}x\,\mathrm{d}y}{\sqrt{(\hat{x}-x)^2+(\hat{y}-y)^2+\hat{z}^2}}\\ &= \sum_{x\in\nabla R}\alpha_R(x)\,\Phi_2(x-\hat{x},y-\hat{y},-\hat{z})\,.\end{aligned} \tag{29}$$

Obviously $|\nabla R| = 4$.

For a linear piece I (located on the x-axis and thus in the planes $z = 0$ and $y = 0$) we have

$$\begin{aligned}\varphi_I(\hat{x},\hat{y},\hat{z}) &:= \int_I \frac{\mathrm{d}x}{\sqrt{(\hat{x}-x)^2+\hat{y}^2+\hat{z}^2}}\\ &= \sum_{x\in\nabla I}\alpha_I(x)\,\Phi_1(x-\hat{x},-\hat{y},-\hat{z})\,.\end{aligned} \tag{30}$$

Of course, in this case I is an interval and the corners are the end-points of this interval, $|\nabla I| = 2$.

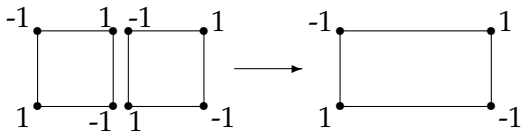


Figure 1: Merging two simple corner complexes

4 Building sources of complex shape from rectangular sources

Computing the electric fields emanating from areal charge patterns on the photo-conductor of an electrophotographic copier was the task from which the present investigation originated. For studying electrophotographic image development (see the introduction of [5],[6] for a short explanation of this) by means of computational models it is desirable to deal with more complex charge patterns than just a single rectangle. Although the underlying industrial project was discontinued before this point became vital, the present work is still inspired by the old problem field. So we will choose a letter (actually the letter F) as a concrete example of a charge distribution. See Figure 2 for the definition of the shape of this model letter. This shape is a bit contrived in that the upper mostly horizontal branch comes finally down to touch the short horizontal branch in just one point. This creates a special situation (expressed by the number 2 assigned to the merging point) which we would not see in a regular letter. As given by the figure, we have an areal charge distribution (2D, in short). Since we aim at treating a spatial example too (3D, in short), we let the charge distribution optionally have a constant vertical extension. With respect to the system of coordinates which is indicated in Figure 2 the 3D charge distribution is confined to the slice $-\mathrm{u}/2 \leq z < \mathrm{u}/2$, whereas the 2D charge distribution is confined to the plane $z = 0$. The height of the letter is $6\,\mathrm{u}$. As is clear from the auxiliary thin lines, the areal shape is a non-overlapping union of the generating square $-\mathrm{u}/2 \leq x, y \leq \mathrm{u}/2$ and 13 shifted copies of it.

Of course, we can consider the potential of the letter as the linear superposition of these 14 patches. This would be computationally waste-full (although trivial to program). A more economical method is the one described in Section 2, culminating in equation (14). It was carried out numerically and the next section describes and shows some results.

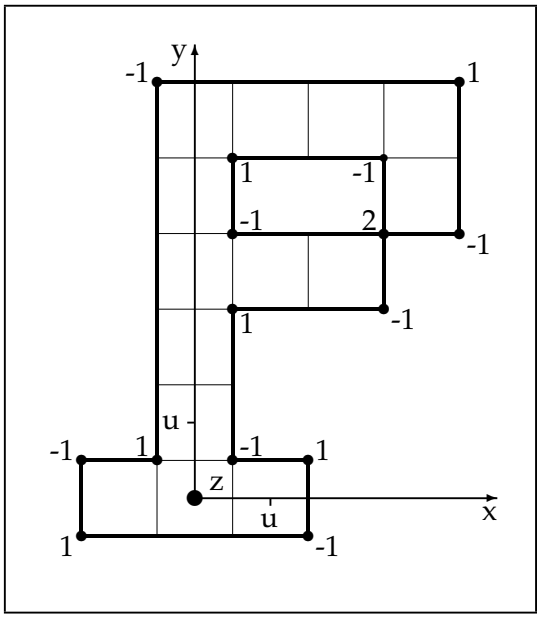


Figure 2: Corner complex of the F-shaped source

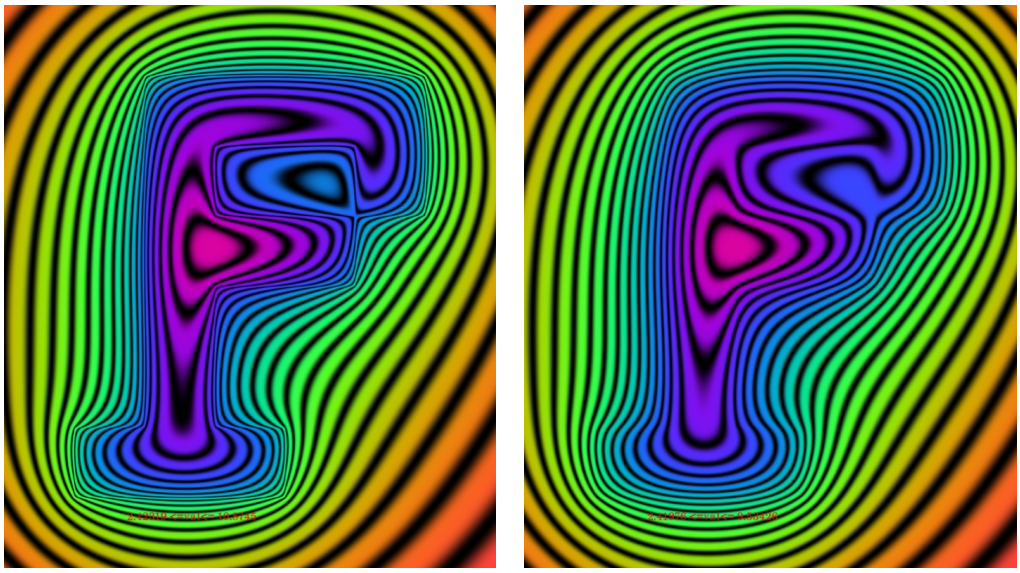
5 Numerical computation of areal potential distributions

Figures 3, 4, and 5 were created with a small C++ program which constructs instances of classes *MultiPatchCharge* and *MultiVolCharge* and calls a few member functions of it. Details can be seen from the first code section in the Appendix. The inner workings of these classes can be seen from the files cpmforcesx.h and cpmforcescpp.h (don't care about the weird file names, there are reasons) for which both listings and compilable code can be freely downloaded from [7]. The most important part of the inner working is the algorithm for the operation '$\uplus$'. It forms the definition of an auxiliary member function ini_() which is called in the constructor code of its class. The code of function ini_() is also given in the Appendix, in the last code section.

The images show the potential always on a planar cross section through the field-filled space. For the images not marked as 'rotated 90 degrees' this image plane is given by $z = 0$, otherwise it is the plane $x = 1$ (the program sets u = 1 in Figure 2). The antiderivative of $1/r$ is regularized at the origin in such a manner that it causes no adverse effects when the imaged plane runs straight through the charge distribution. The value range of the potential is transformed into a range of color hues according to 'photon energy coding': violet indicates the highest values and red the lowest ones, in between are the colors in spectral order. To improve our ability to compare levels there is also a very distinct brightness modulation that creates dark 'Fraunhofer lines' which connect points of approximately the same potential.

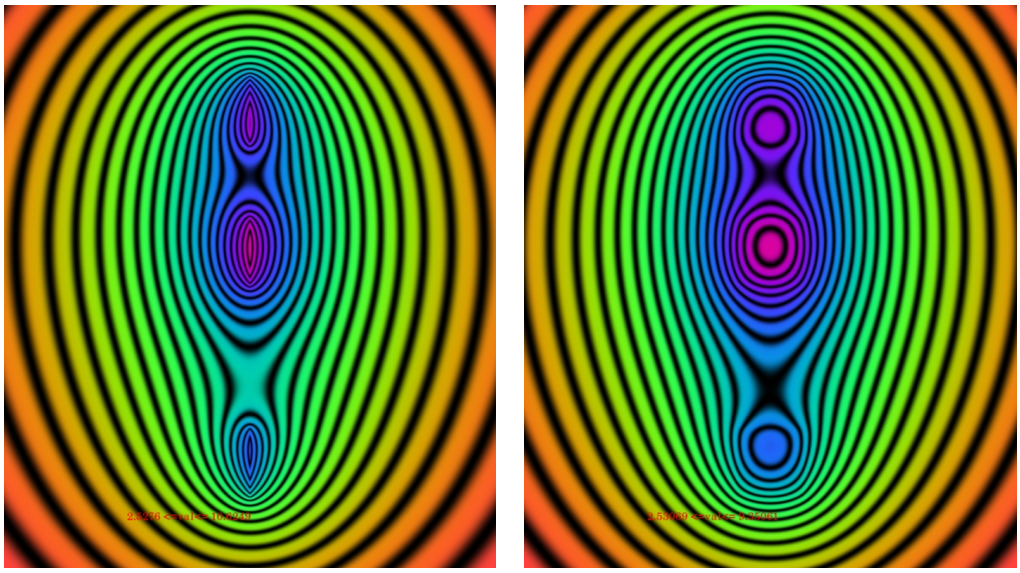
Figure 5 with mirror charges shows the potential distribution as it holds for the electrostatic image in a copier. Here we have a conducting layer in the photo-conductor loop the influence of which can be modeled by introducing a charge inverted system at the mirrored positions with respect to the conducting layer. Fortunately we obtain the corner complex of this system simply by shifting the corners of the original system. Actually classes MultiPatchCharge and MultiVolCharge are defined such that each instance has its own 'personal' system of reference which controls all interactions of the objects with the out-side world. Therefore, all geometric transformations (including rotations!) can be implemented by only transforming this personal system of reference.

Computing the potential at 700×800 points (one point per image pixel, no interpolation) and storing the pixel data in RAM took 1.9 s on an off the shelf 2.4 GHz laptop computer. This says that one pixel consumes 8100 processor cycles.



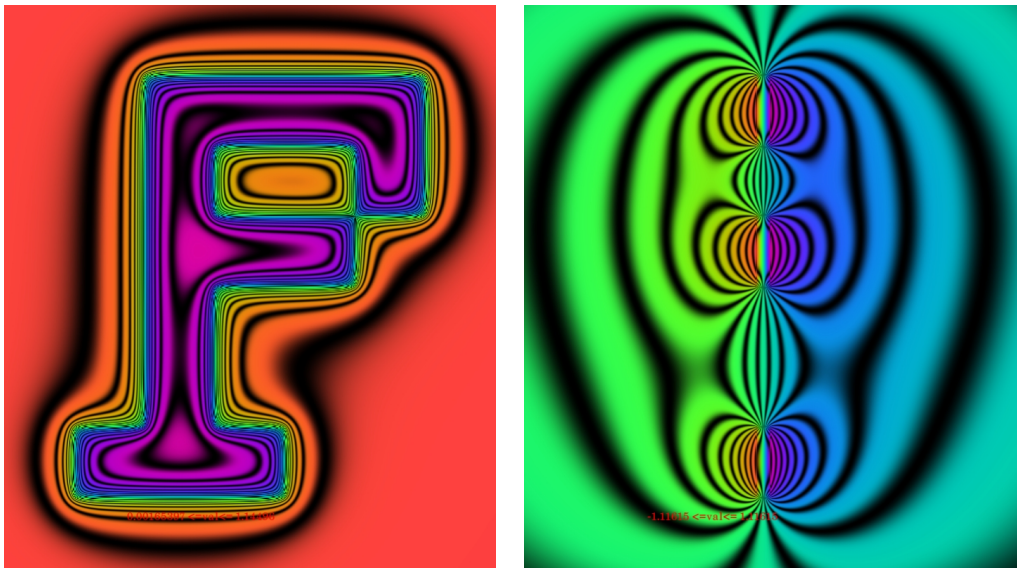
(a) 2D (b) 3D

Figure 3: Potential distributions of a F-shaped source



(a) 2D (b) 3D

Figure 4: Potential distributions of a F-shaped source rotated 90 degrees



(a) normal view (b) rotated 90 degrees

Figure 5: Potential distributions of a F-shaped source including mirror charges

6 Appendix

This is the code that created the figures 3, 4, 5. Input is expected from file temptest.ini to be shown subsequently. The classes (or types) occurring in the code are: `Frame`: graphical window, `Camera`: universal visualization tool (also stereo), `Word`: character string, `SysControl`: general control parameters, `R`: real numbers, `RefSys`: reference system in space, `Vec`: geometrical vector, `Iv`: interval, `R2`: pairs of real numbers, `R3`: triplets of real numbers, `V`: vector template, `ScalarField`: scalar field, `MultiPatchCharge`: corner complexes which result from a single 2D rectangle and a family of translation vectors to build complex shapes in a plane. `MultiVolCharge`: corner complexes which result from a single 3D rectangle and a family of translation vectors to build complex shapes in space. These last two classes were added to the *CPM* files cpmforcesx.h and cpmforcescpp.h for the purpose of the present investigation. All the other classes belong to the *CPM* file system [7] (free download) since years.

```
#define CPM_SC\
   vr[1]=e2;\
   vr[2]=vr[1]+e2;\
   vr[3]=vr[2]+e2;\
   vr[4]=vr[3]+e1;\
   vr[5]=vr[3]+e2;\
   vr[6]=vr[5]+e2;\
   vr[7]=vr[6]+e1;\
   vr[8]=vr[7]+e1;\
   vr[9]=vr[8]+e1;\
   vr[10]=vr[9]-e2;\
   vr[11]=vr[4]+e1;\
   vr[12]=vr[1]-e2+e1;\
   vr[13]=vr[1]-e2-e1;
      // this is the definition of the 'F', uses the shared code macro
      // for avoiding code duplication

void CpmTests::TempTestAppl::doTheWork()
{
   cpmmessage("TempTestAppl::doTheWork() started");
   Frame fr;
   Word sec("cameracontrol");
   Camera ca(rch,sec,fr); // rch is an instance of class RecordHandler
      // which holds all data contained in file temptest.ini
   ca.clr();
   sec="syscontrol";
   SysControl::read(rch,sec);
   sec="data";
   R tWait=10, dz=0;
   bool areal=true;
   cpmrh(tWait); // 'read handler' general input macro
   cpmrh(areal);
   cpmrh(dz);
   RefSys rs; // Euclidean reference system in 3-space
   Vec shift(V<R>("",0,0,dz));
   R sigma=1./CpmPhysics::influence_PHYS; // Then ePot gets multiplied by
      // influence_PHYS. So the value of this physical constant cancels
      // out, as intended here.
   R r=1e-4;
   Iv iv1(-0.5,0.5);
   Iv iv2=iv1;
   R s1=iv1.abs();
   R s2=iv2.abs();
```

```
   if (areal){
      R2 e1(s1,0);
      R2 e2(0,s2);
      V<R2> vr(13);
      CPM_SC
      MultiPatchCharge mpc(iv1, iv2, rs, sigma,r,vr);
      ScalarField pot=mpc.fieldU();
      if (dz!=0){ // then add mirror charges
         MultiPatchCharge mpc2(mpc); // cheap duplication
         mpc2.invChr_(); // invert charge
         mpc2+=shift; // move the mirror charges to a parallel plane
         pot+=mpc2.fieldU(); // scalar fields may be added
      }
      pot.mark(ca);
   }else{
      Iv iv3=iv1;
      R s3=iv3.abs();
      R3 e1(s1,0,0);
      R3 e2(0,s2,0);
      R3 e3(0,0,s3);
      V<R3> vr(13);
      CPM_SC
      MultiVolCharge mpc(iv1, iv2, iv3,rs, sigma,r,vr);
      ScalarField pot=mpc.fieldU();
      if (dz!=0){ // then add mirror charges
         MultiVolCharge mpc2(mpc); // cheap duplication
         mpc2.invChr_();
         mpc2+=shift;
         pot+=mpc2.fieldU();
      }
      pot.mark(ca); // creates the image data
   }
   ca.display(true); // creates image file in ppm format
   cpmwait(tWait,3); // gives time to inspect the image on screen
   cpmmessage("TempTestAppl::doTheWork() done");
}
#undef CPM_SC
```

This is the file with the control data which determine the actual computation. Notice that input is associates with a type (one of B, Z, R, W, Bs, Zs, Rs, Ws), a name (e.g. tWait) and a section (e.g. data).

```
// temptest.ini
about
//------------------------------------------------------------------------
   W file=temptest.ini
   Ws line1=Testing class MultiPatchCharge
   Ws line2=and class MultiVolCharge

selection
//------------------------------------------------------------------------
Z sel=1
   // selection; if not zero, a documentation file will be created

Z cpmverbose=1
   // controls the degree of detail with which program run
   // information will be written to log file cpmcerr.txt

Z cpmdbg=2
   // 0: asserts ignored, 1: failing asserts warn, 2: failing asserts stop
```

```
W runName=100725b
   // run identifier; gets appended to documentation and result files

Z mode=-1
   // not needed, no influence as long as negative

      .....

data
//-------------------------------------------------------------------------
R tWait=12
B areal=true
R dz=0.2
   // if this is not 0 there are mirrorcharges at that distance

cameracontrol
//-------------------------------------------------------------------------
   R aspectRatio=0.875
   R fieldOfViewCenterX=1.0
   R fieldOfViewCenterY=2.5
   R fieldOfViewCenterZ=0.0
   R fieldOfViewHeight=8
   R panDeg=90
      // 0 and 90
   R rollDeg=0.0
   R shiftX=0.0
   R shiftY=0.0
   R tiltDeg=0
   R viewingAngleDegree=30
   Z yPixel=800
      // here equal to nyScalarFields, thus no interpolation
   Z responseMode=6
   R responseGamma=-1

syscontrol
//-------------------------------------------------------------------------
   Z nyScalarFields=800
      // number of points in y-direction in which
      // scalar fields are sampled for visualization
```

Coding the algorithm which defines operation '⊎'

```
/////////////// class MultiPatchCharge ////////////////////////////////////

void MultiPatchCharge::ini_(void)
{
   S<R2> sr; // set of R2
   M<R2,Z> mrz; // map with keys of type R2 and values of type Z
   Z i; // Z integer numbers
   R2 c1(xr_[1],yr_[1]); // xr_: x-range of reference patch
   R2 c2(xr_[1],yr_[2]); // yr_: y-range of reference patch
   R2 c3(xr_[2],yr_[1]); // names of data members end in '_'
   R2 c4(xr_[2],yr_[2]);
   sr.add(c1); sr.add(c2); sr.add(c3); sr.add(c4);
   R2 cc1=c1, cc2=c2, cc3=c3, cc4=c4;
   Z z1=1, z2=-1, z3=-1, z4=1;
   mrz[cc1]=z1;
   mrz[cc2]=z2;
   mrz[cc3]=z3;
```

```
   mrz[cc4]=z4;

   bool bi;
   for (i=v_.b();i<=v_.e();++i){ // v_ is an array of translation vectors
      R2 vi=v_[i];
      cc1=c1+vi;
      cc2=c2+vi;
      cc3=c3+vi;
      cc4=c4+vi;
      bi=sr.addAct(cc1); // returns true if cc1 is not yet in sr
      if (bi) mrz[cc1]=z1; else mrz[cc1]+=z1;
      bi=sr.addAct(cc2);
      if (bi) mrz[cc2]=z2; else mrz[cc2]+=z2;
      bi=sr.addAct(cc3);
      if (bi) mrz[cc3]=z3; else mrz[cc3]+=z3;
      bi=sr.addAct(cc4);
      if (bi) mrz[cc4]=z4; else mrz[cc4]+=z4;
   }
   for (i=mrz.b();i<=mrz.e();++i){ // purging corners with weight 0
      R2 ci=mrz.x(i);
      Z  zi=mrz.y(i);
      if (zi!=0){
         sx_.add(X2<R2,Z>(ci,zi));
      }
   }
}
```

Acknowledgment

I thank Amir Finkelstein for extended discussions on the discrete Green's theorem and his concepts of detachment and tendency.

References

[1] Ulrich Mutze: The Fundamental Theorem of Calculus in $\mathbb{R}^n$
Mathematical Physics Preprint 04–165, May 2004
www.ma.utexas.edu/mp_arc/c/04/04-165.pdf

[2] Xiaogang Wang, Gianfranco Doretto, Thomas Sebastian, Jens Rittscher, Peter Tu: Shape and Appearance Context Modeling. In Proc. IEEE Int. Conf. on Computer Vision (ICCV), pages 1-8, 2007
http://vision.ucla.edu/~doretto/publications/wangDSRT07iccv.pdf

[3] Amir Finkelstein: The Theory Behind The "Summed Area Tables" Algorithm: A Simple Approach To Calculus. arXiv:1005.1418v3[cs.DM] May 2010

[4] Amir Finkelstein: Slanted Line Integral, The Wolfram Demonstrations Project
http://demonstrations.wolfram.com/SlantedLineIntegral/

[5] Ulrich Mutze: Rigidly connected overlapping spherical particles: a versatile grain model, Granular Matter 2006

[6] Ulrich Mutze: Polyspherical grains and their dynamics
Mathematical Physics Preprint 07–252, July 2007
`www.ma.utexas.edu/mp_arc/c/07/07-252.pdf` (2007)

[7] Ulrich Mutze: Classes for Physics and Mathematics (also called *CPM* or C+−)
`www.ulrichmutze.de/softwaredescriptions/cpmlisting.pdf`
`www.ulrichmutze.de/softwaredescriptions/cpmProject.pdf`
`www.ulrichmutze.de/softwaredescriptions/tut.pdf` (C+− tutorial)
`www.ulrichmutze.de/sourcecode/cpm.zip`